LBL-12054
c2

NRCC NATIONAL RESOURCE FOR COMPUTATION IN CHEMISTRY

Presented at the Second Chemical Congress of the North American Continent, Las Vegas, NV, August 24-29, 1980

THE RELATIVE PERFORMANCES OF SEVERAL SCIENTIFIC COMPUTERS FOR A LIQUID MOLECULAR DYNAMICS SIMULATION

D.M. Ceperley

August 1980

LBL-12054 c2

LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA

Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48 and for the National Science Foundation under Interagency Agreement CHE-7721305

LBL-12054

The Relative Performances of Several Scientific Computers for a Liquid Molecular Dynamics Simulation

By

D. M. Ceperley
National Resource for Computation in Chemistry
Lawrence Berkeley Laboratory, University of California,
Berkeley, CA 94720

In the last decade, the computer modeling of matter by simulations has become a very important area of theoretical chemistry.(1) One goal of the simulations is to understand the properties of macroscopic systems starting from the Coulomb potential and Schroedinger equation. Although it is feasible that simulation methods can treat the complete many-body quantum problem,(2) most simulations today assume a classical model with some effective interparticle potential. There are two common methods employed, Metropolis Monte Carlo (MC)(3) is an effective algorithm used for calculating static properties of many-body systems. Molecular Dynamics (MD)(4) is the term employed when Newton's equations of motion are solved to find equilibrium and non-equilibrium, dynamic and static properties of many-body systems. What all of these simulations methods have in common and their limitation is a processor fast enough to move hundreds of atoms, hundreds and thousands of times. What I want to discuss in this short note, are some of the computational characteristics of simulations and my experience in using a standard simulation program on several scientific computers.

While at the National Resource for Computation in Chemistry, I have developed a general classical simulation program, called, CLAMPS (for classical many particle simulator)(5) capable of performing MC and MD simulations of arbitrary mixtures of single atoms. The potential energy of a configuration of N atoms at positions $R = \{r_1, \ldots, r_N\}$ and with chemical species $\{\alpha_1, \ldots, \alpha_N\}$ is assumed to be a pairwise sum of spherically symmetric functions.

$$U = \sum_{i<j} \phi_{\alpha_i \alpha_j}(|r_i - r_j|_M) \qquad (1)$$

Where $\phi_{\alpha\beta}(r)$ is the interaction between two atoms of type α and β and $|r|_M$ means the minimum image distance consistent with periodic boundary conditions. In addition, there can be bonding potentials between certain pairs of atoms. If some of the atoms are charged there is another term in the potential energy arising from the interaction of a charge with the charges outside the simulation box: the Ewald image potential.(6) This can be conveniently written as

$$U_E = \sum_k v_k |\rho_k|^2 \tag{2}$$

Where k is a vector in the reciprocal lattice of the simulation cell, ρ_k is the Fourier transform of the charge density,

$$\rho_k = \sum_i q_i \exp(ik \cdot r_i) \tag{3}$$

q_i is the charge of particle i, and v_k is the Fourier component of the long range potential.(6)

In simulations, computation of the potential energy and forces takes the vast majority of the computer time. The other operations, such as moving the particles, usually are much quicker. Shown in Table I is the FORTRAN coding needed to compute the pair sum in eq. (1). In a general purpose program such as CLAMPS one cannot assume that the pairwise interactions are simple enough to compute at each step, whereas, a table lookup is equally efficient for all systems. In CLAMPS the potentials and the derivative $-r^{-1}d\phi/dr$ are computed on a grid linear in r^2 at the beginning of the program and stored. Tables with the order of 10^4 entries usually give sufficient accuracy for most problems without any interpolation because of the statistical nature of the computation.

The coding in Table I illustrates the central problem of simulations. The number of pairs is N(N-1)/2. The number of floating point operations (FLOPS) per pair is about 25, assuming the branches are executed 50% of the time. Thus for 100 atoms (a minimal simulation) we will need 1.2×10^5 FLOPS for a single time step. The number of memory and indexing operations is similarly large. Typically one needs to execute between 10^3 and 10^5 time steps. Thus the simulations are limited by the number of floating point operations one can afford.

For systems which can be modeled with particles interacting with only short ranged forces (that is the potential can be neglected beyond several neighbor shells), the number of operations per time step will be proportional to the number of particles times the average number of neighbors of a given particle. For such models, simulations of 10^4 atoms are possible today on available mainframe as well as minicomputers. For many chemical systems, such as those containing macromolecules, one would like to work with still larger systems over many time steps. Even with today's computers, most chemical systems cannot be simulated without

Table I

```
C LOOP OVER ALL PAIRS OF ATOMS I, J
      DO 1 I=1, NATOMS-1
      DO 2 J=I+1, NATOMS
C CALCULATE PERIODIC DISTANCES
      R2=0.0
      DO 3 L=1, NDIM
      DX(L)=X(I,L)-X (J,L )
C ELL AND EL2 ARE THE BOX AND HALF THE BOX LENGTHS
      IF (DX(L).GT.EL2(L)) DX(L)=DX(L)-ELL(L)
      IF (DX(L).LT.-EL2(L)) DX(L)=DX(L)+ELL (L)
3     R2=R2+DX( L )**2
C IT AND JT ARE THE CHEMICAL TYPES
      IT=ITYPE( I )
      JT=ITYPE( J )
C CONVERT DISTANCE TO A TABLE ENTRY
      LI=CSI ( IT,JT)*R2
C IF OUTSIDE TABLE POTENTIAL IS ZERO
      IF(LI.GE.LMAX) GO TO 2
C LT IS THE TABLE FOR THIS INTERACTION
      LT=LTABLE( IT,JT)
C LOOK UP POTENTIAL AND DERIVATIVE
      V=V+EPS(IT,JT)*PTABLE(LI,LT)
      FT=EPSF(IT,JT)*FTABLE(LI,LT)
C NOW ADD TO FORCES
      DO 4 L=1, NDIM
      F=FT*DX(L)
      FORCE( I,L )=FORCE(I,L)+F
4     FORCE( J,L )=FORCE(J,L)-F
2     CONTINUE
1     CONTINUE
```

FORTRAN Code for the pairwise sum of eq. (1).

making many simplifying assumptions. Both a supercomputer as well as better algorithms are necessary to tackle these problems.

For the purpose of comparing performance on different computers, I have used the Stillinger-Lemberg(7) model for water. This model contains central force interactions between charged oxygen and hydrogen atoms. The three different potential functions between OO, OH and HH, are tailored to give the correct geometry and dipole moment for an isolated molecule and some of the pair bonding properties of two molecules

Simulations of charged systems are very important. Common examples are plasmas, ionic solutions, dipole system and electronic systems. Because all pairs are included in the sum of eq. (2), the computer time only depends on the number of atoms and the number of time steps. For this reason my results should be applicable to all similar systems. I will discuss here only results for molecular dynamics simulations. The situation for Monte Carlo is completely parallel, although the actual coding is different since atoms are moved singly rather than all together.

Because the atoms are charged, the Ewald image potential from eq. (2) must be used to account for the long-range Coulomb potential. In the following benchmarks, I have included all terms in the sum in eq. (2) for which $k \leq 6\pi/L$; this comprises 123 terms, and is adequate to represent the potential to one part in 10^4. As long as the number of terms is held fixed, the computer time to evaluate eq. (2) will be proportional to the number of atoms while the pairwise sum in eq. (1) will take time proportional to the square of the number of atoms. Thus for large enough systems, it is the pairwise sum which dominates the calculation. The sines and cosines needed for ρ_k are computed recursively. I will not discuss the computation of the Ewald sum in detail, because it is relatively specialized.

Computer Comparisons

In this section, I will discuss the programming considerations and timing results for the four computers on which I have tested CLAMPS. In all cases the code was not substantially changed. Essentially only the routines which performed the sums in eqs. (1) and (2) were modified. All changes were in FORTRAN or with FORTRAN callable routines. The timing results are not optimal, but rather typical of what could be achieved by a user in FORTRAN. The timing results are given in Table II for systems containing 27 and 216 molecules (81 and 648 atoms). MFLOPS refers to the number of million floating point operations per second in executing the pairwise sum of Table I assuming each pass through consists of 25 floating point operations.

Table II

T_p is the time in seconds to execute the pairwise sum in equation (1); T is the total time in seconds per molecular dynamic step. MFLOPS is the number of million floating point operations per second of the code in Table I, assuming that it contains 25 FLOPS (i.e., MFLOPS = $1.25 \times 10^5 \times N(N-1)/T_p$) where N is the total number of atoms, 81 or 648. The asterisk on CRAY-1 indicates a vectorized version of CLAMPS was used.

Computer	81 Atoms			648 Atoms		
	Tp	MFLOPS	T	Tp	MFLOPS	T
VAX 11/70	0.35	0.23	1.63	22.2	0.24	32.5
CDC 7600	0.033	2.5	0.125	2.1	2.5	2.85
CRAY-1	0.0182	4.5	0.100	1.1	4.8	1.77
CRAY-1*	0.0070	11.6	0.0157	0.257	20.4	0.311
VAX-FPSAP	-	-	-	25.0	0.21	-

DEC VAX 11/70

The VAX used, is located at NRCC in Berkeley, has a floating point accelerator, 2.5 M Bytes of memory, and was running version 1.3 of the operating system. The code was run in single precision (32 bits/word) and that was found adequate to conserve energy and give satisfactory equilibrium properties. The code used to perform the pairwise sum is essentially that of Table I.

CDC 7600

The 7600 used is located at Lawrence Berkeley Laboratory, is approximately ten years old and has 65 K of 60 bit word fast memory (small core). Because CLAMPS has dynamic memory allocation, it is possible to fit a simulation in fast memory of up to about 2000 atoms as long as the potential tables are not too extensive. The compiler used was the standard CDC FTN 4.8, OPT=2. The only difference between the CDC coding of the pairwise sum and that in Table I is that the periodic boundary conditions (loop 3) are handled by Boolean and shift operations instead of branches. Branches on the 7600 causes all parallel processing to halt.

CRAY-1

The CRAY used is located at Lawrence Livermore Laboratory. Characteristics of the CRAY are described elsewhere in this volume. There is a large advantage in achieving vector rather then scalar code. This can be seen in Table II. Initially, the CDC version of CLAMPS was run on the CRAY with the time results showing it only slightly faster than the 7600. Several subroutines of CLAMPS were then vectorized and the simulation executed in approximately 1/5 the time. The vectorized version of the pairwise sum appears in Table III. The problems encountered in vectorizing this routine were:

1) The periodic boundary conditions in loop 3 contain 2 branches. Vectorization was achieved by using the FORTRAN callable vector merge function.

2) The branch for the case when the squared pair separation is outside the table will inhibit vectorization. The last element of the table has been changed to zero and all occurrence outside the table are truncated to LMAX. The rest of the code, which is not executed on the VAX or CDC 7600, is executed here. It is often necessary on a vector machine to increase the total number of floating point operations to achieve vector rather than scalar processing. The MFLOP rates reported here are computed on the basis of the original number of floating point operations. The extra ones added to achieve vectorization are not included.

3) The table look-ups for the force and potential can be done with the GATHER function. GATHER (N, A, B, INDEX) is equivalent to the FORTRAN statements.

```
          DO 2 I = 1, N
    2     A(I) = B(INDEX(I))
```

Although GATHER is a scalar operation and rather slow, 12 machine cycles/element, (a machine cycle is 12.5 ns), GATHER is faster than computing all but the simplest inverse power potentials. By comparison a square root takes 14 machine cycles/element and the entire pairwise sum takes an average of 98 machine cycles/pair. Note that temporaries are set up for the scaling factors of the potential, as well as the addresses for the start of the tables. These temporaries are changed only rarely and so do not affect the timing. They would be unnecessary if there were only one type of particle.

Table III

```
C LOOP OVER ALL PAIRS OF ATOMS I,J
      ITL=0
      DO 1 I=1,NATOMS-1
      I1=I+1
      NC=NATOMS-I
C CHECK TO SEE IF WE NEED TO REFRESH OUR TEMPORARIES
      IF( ITYPE(I).EQ.ITL) GO TO 20
      ITL=ITYPE( I )
C GATHER MAKES EPST(J )=EPS(ITYPE(J),ITYPE(I))
      CALL GATHER(NC,EPSF (I1),EPS(1,ITL),ITYPE(I1))
C GATHER ALSO EPSF,  CSI AND LTABLE
      CALL GATHER(NC,EPSFT(I1),EPSF(1,ITL),ITYPE(I1))
      CALL GATHER(NC,CSIT(I1),CSI(1,ITL),ITYPE(I1))
      CALL GATHER(NC,LTABT(I1),LTABLE(1,ITL),ITYPE(I1))
2     DO 3 J=I1,NATOMS
3     R2(J)=0.
C CALCULATE PERIODIC DISTANCES
      DO 4 L=1,NDIM
      T=SIGN( ELL(L),X(I,L)
      DO 5 J=I1,NATOMS
C CVMGP(X,V,Z)=X If Z.GT.0 AND Y OTHERWISE
5     DX(J,L )=CVMGP((X(I,L)-X(J,L))-T,X(I,L)-X(J,L)
      +,ABS(X( I,L )-X( J,L ))-EL2( L ))
      DO 4 J=I1, NATOMS
4     R2(J)=R2(J)+DX(J,L)**2
C MAKE R2 INTO TABLE ENTRIES WITH LMAX BEING MAXIMUM
      DO 6 J=I1,NATOMS
6     INDEX(J)=LTABT(J)+MINO (LMAX,INT(CSIT(J)*R2(J)))
C GATHER V(R( J ) )-INTO A VECTOR
      CALL GATHER(NC,R2(I1),PTABLE, INDEX(I1)
C SUM THEM UP
      VTOTAL=VTOTAL+SDOT(NC,R2(I1),1,EPST(I1),1)
C GATHER DERIVATIVES FROM FTABLE INTO R2
      GALL GATHER(NC,R2(I1),FTABLE,INDEX(I1),1)
      DO 7 J=I1, NATOMS
7     R2(J)=R2(J)*EPSFT(J)
C MULTIPLY BY DISPLACEMENTS AND ADD INTO FORCE VECTORS
      DO 8 L=1, NDIM
      FORCE (I,L)=FORCE(I,L)+SDOT(NC,R2(I1),1,DX(I1,L),1)
      DO 8 J=I1,NATOMS
8     FORCE(J,L)=FORCE(J,L)-R2(J)*DX(J,L)
1     CONTINUE
```

FORTRAN code, optimized for the CRAY, which performs the pairwise sum of eq. (1).

4) The summing of the force on particle I can be performed by the BLAS (8)(Basic Linear Algebra Subroutine) SDOT, which is quite efficient (about 3.5 machine cycles/element).

VAX 11-70 with attached Floating Point Systems Array Processor (120B)

The system used is that located in the Chemistry Department at Columbia University. The architecture and characteristics of the Array Processor (120B) are described elsewhere in this volume. A very careful investigation of the use of an array processor to perform Monte Carlo simulations has been done by Chester, et. al.(9) There they demonstrated that with assembly language hand coding (APAL) a simulation on the AP would run about twice as long as an CDC 7600. Here we report the timing results for the pairwise sum using APTRAN, the FORTRAN Cornell computer (Level 3.5). Because of the size of CLAMPS (3500 lines), and the intermix between calculations and I/O operations, the entire code cannot be compiled in the AP. Instead a single subroutine, which contains the coding in Table I was written, with all data passed through a common block. This routine then executes on the AP with the rest of the program executing on the host (VAX 11-70). Before each calculation of the pairwise sum, the coordinates need to be passed to the AP. After it is finished the potential and forces are then passed back to the host. The transmittal time is small (less then 10 ms), so that this mode of operation is satisfactory if there are at least 100 particles. However, the code generated by the compiler is not nearly as efficient as assembly language coding would be, the pairwise sum executes in roughly the same time as in the VAX alone. Undoubtedly restructuring of the FORTRAN would improve the execution time. The accuracy (38 bits) of the AP is sufficient for molecular dynamics. A general review of array processors and their usefulness for chemical computations is contained in Ref. (10).

Simulation Needs

Finally I would like to summarize the computational characteristics of simulations and what makes a good simulation computer.

1) A simulation is almost always cpu bound. Hence, fast floating point speed is essential.

2) Memory size can be made quite small, 20K to 200K.

3) Accuracy demands are less than in many areas of computational chemistry. Usually one needs accuracy only to one part in 10^4 in energy for Monte Carlo and one part in 10^6 for the forces in Molecular Dynamics.

4) As we have seen on the CRAY the ability to gather data together is essential. Memory speed must be commensurate with floating point speed. When nearest neighbor tables are used fast scatter operations are also needed. The two essential random memory operations needed are:

$$B(I) = A(INDEX(I))$$
$$B(INDEX(I)) = B(INDEX(I)) + A(I)$$

5) Except for the above gather-scatter operations, simulations are easily vectorizable as defined by the CRAY FORTRAN or the CYBER 200 FORTRAN. Typical vectors have 50 to 500 elements each.

6) The Basic Linear Algebra Subroutines (BLAS) are a convenient way of maintaining efficiency and portability. They should be extended to include such things as GATHER, and vectorized EXP, SQRT, SIN and COS.

Acknowledgment

This research was supported in part by the Office of Basic Energy of Science of the U. S. Department of Energy under Contract No. W-7405-ENG-48 and by the National Science Foundation under Grant No. CHE 7721305. Special thanks to Dr. M. Rao and Prof. B. Berne for making available and assisting me in the use of the computing facilities in the Chemistry Department at Columbia University.

Literature Cited

1. Lykos, P., Ed. "Computer Modeling of Matter"; ACS Symposium Series 86: Washington, D. C., 1978.

2. Ceperley, D.; Alder, B. J.; Phys. Rev. Letts. 1980, 45, 566.3. Metropolis, M.; Rosenbluth, A. W.; Teller, M. N.; Teller, E.; J. Chem. Phys. 1953, 21, 1087.

4. Hansen, J. P.; McDonald, I. R.; "Theory of Simple Liquids"; Academic Press: New York, 1976; Chapter 3.

5. Contact the Quantum Chemistry Program Exchange, Department of Chemistry, Room 204, Indiana University, Bloomington, Indiana, 47405 for CLAMPS.

6. Valleau, J. P.; Whittington, S. G.; "Modern Theoretical Chemistry 5A"; Ed. B. Berne; Plenum: New York, 1977.

7. Rahman, A.; Stillinger, F.; Lemberg, H.; J. Chem. Phys. 1975, 63, 5225.

8. The BLAS are a collection of 38 FORTRAN callable subroutines that peform many of the basic operations of numerical linear algebra. Contact International Mathematical and Statistical Libraries, (IMSL) for more information.

9. Chester, G.; Gann, R.; Gallagher, R.; Grimson, A.; "Computer Modeling of Matter" Ed. P. Lykos; ACS Symposium Series 86: Washington, D. C., 1978.

10. Ostlund, N. S.; "Attached Scientific Processors for Chemical Computations: A report to the Chemistry Community", NRCC report (1980).